

Algoritmos y Estructuras de Datos II

Primer parcial – Lunes 19 de septiembre de 2022

- Es posible tener una hoja (2 carillas), escrita a mano, con los apuntes que se deseen
- Cada ejercicio debe entregarse en **hojas separadas**. Incluir en cada hoja su Nro LU, número de hoja, apellido y nombre.
- Cada ejercicio se calificará con **Aprobado, Regular, o Insuficiente**.
- El parcial estará aprobado si las notas de los tres ejercicios son mejores o iguales a: {R, A, R}, {R, R, A} o {I, A, A}.

Ej. 1. Árboles binarios



(A)

!! Felicitaciones!

Extender el TAD $\text{ÁRBOL BINARIO}(\alpha)$, definido en el apunte de TADs Básicos, para que tenga las operaciones listadas más abajo. Definir, además, el tipo de cada operación y sus restricciones. En este ejercicio no está permitido hacer pattern matching con los generadores del tipo (es decir, no puede escribirse **nil** ni **bin** en el lado izquierdo de las axiomatizaciones).

- (a) **máximo**: que toma un árbol de naturales y devuelve el valor máximo de sus nodos.
Se puede suponer para este ejercicio que están definidas las operaciones $\text{max} : \text{nat} \times \text{nat} \rightarrow \text{nat}$ y $\text{max3} : \text{nat} \times \text{nat} \times \text{nat} \rightarrow \text{nat}$ que obtienen el máximo entre dos números y el máximo entre 3 números.
- (b) **sumar**: que dados dos árboles binarios de naturales devuelve un árbol que contenga la suma de sus elementos. En el caso de que uno de los árboles no contuviera un nodo en la misma posición que el otro, el valor será el del árbol que lo tenga definido.

Por ejemplo, dados los siguientes árboles:

$A = \text{bin}(\text{bin}(\text{nil}, 2, \text{nil}), 3, \text{bin}(\text{nil}, 0, \text{nil}))$

$B = \text{bin}(\text{bin}(\text{bin}(\text{nil}, 15, \text{nil}), 3, \text{nil}), 10, \text{nil})$

$\text{máximo}(A)$ debe resultar en 3, mientras que $\text{máximo}(B)$ debe resultar en 15.

$\text{suma}(A,B)$ debería resultar en: $\text{bin}(\text{bin}(\text{nil}, 15, \text{nil}), 5, \text{nil}), 13, \text{bin}(\text{nil}, 0, \text{nil}))$

Ej. 2. Axiomatización

Sea la siguiente sección, parte de la definición del TAD **Stock**:

generadores

$\text{abrirTienda} : \rightarrow \text{stock}$

$\text{nuevoProducto} : \text{stock} \times \text{producto} \times \text{nat umbral} \rightarrow \text{stock}$

$\text{compra} : \text{stock } s \times \text{producto } p \times \text{nat cantidad} \rightarrow \text{stock}$

$\text{venta} : \text{stock } s \times \text{producto } p \times \text{nat cantidad} \rightarrow \text{stock}$

$\{ \# \text{disponibles}(s) + \# \text{disponiblesSust}(p, s) \geq \text{cantidad} \}$

def $\text{sustituto} : \text{stock } s \times \text{producto } p \times \text{producto } \text{sust} \rightarrow \text{stock}$

$\{ p \in \text{productos}(s) \wedge \text{sust} \in \text{productos}(s) \wedge p \neq \text{sust} \wedge \neg \text{tieneSustituto?}(p, s) \}$

$\{ \wedge \neg (\exists p' : \text{producto}) p' \in \text{productos}(s) \wedge \text{sustituto}(p', s) = \text{sust} \}$

observadores básicos

$\text{productos} : \text{stock} \rightarrow \text{conj}(\text{producto})$

$\text{umbral} : \text{stock } s \times \text{producto } p \rightarrow \text{nat}$

$\text{tieneSustituto?} : \text{producto } p \times \text{stock } s \rightarrow \text{bool}$

$\text{sustituto} : \text{producto } p \times \text{stock } s \rightarrow \text{producto}$

$\# \text{disponibles} : \text{producto } p \times \text{stock } s \rightarrow \text{nat}$

$\{ p \in \text{productos}(s) \}$

$\{ p \in \text{productos}(s) \}$

$\{ p \in \text{productos}(s) \wedge \text{tieneSustituto?}(p, s) \}$

$\{ p \in \text{productos}(s) \}$

otras operaciones

$\text{bajoUmbral} : \text{stock } s \rightarrow \text{conj}(\text{producto})$

$\# \text{disponiblesSust} : \text{stock } s \times \text{producto } p \rightarrow \text{nat}$

$\{ p \in \text{productos}(s) \}$

Es decir, STOCK permite la siguientes funcionalidades:

- Registrar un nuevo producto con su umbral mínimo de reposición (un natural).
- Registrar las compras y las ventas de un producto, indicando la cantidad de elementos comprados y vendidos.
- Permitir que un producto tenga a lo sumo un sustituto y que, a su vez, este no sea sustituto de más de un producto.
- Al realizarse una venta, en caso que no haya suficiente cantidad del producto, pero sí de su sustituto, se reemplazará automáticamente por la venta de la misma cantidad del producto sustituto.
- Permitir devolver el conjunto de los productos tales que el total del stock del producto sumado a el de su sustituto esté por debajo del umbral mínimo de reposición.

Se pide:

- (a) Completar la axiomatización de los observadores `tieneSustituto?`, `sustituto` y `#disponibles`.
- (b) Completar la axiomatización de las funciones `bajoUmbral`, `#disponiblesSust`.

Ej. 3. Modelado

Se quiere resolver la asignación de aulas del nuevo edificio $0 + \infty$ para primer cuatrimestre del año que viene. Así, las materias podrán solicitar las aulas que van a necesitar y se realizará la asignación correspondiente teniendo en cuenta la siguiente información:

- El departamento cuenta con una serie de franjas horarias fijas a lo largo de todo el cuatrimestre. Por ejemplo “la franja de los miércoles de 17:00 a 22:00”, “la franja de los viernes de 10:00 a 12:00”, etc. Además, se conoce el grupo de materias y el de aulas que tiene la facultad. Para el caso de las aulas, se conoce su capacidad máxima, y se sabe que no hay dos que puedan albergar la misma cantidad de personas.
- El sistema permite que una materia haga un pedido de reserva de aula, especificando la franja particular que desea y la cantidad de estudiantes esperada. Una materia puede hacer más de un pedido para distintas franjas.
- El pedido se concretará o se rechazará de forma automática, dependiendo de si se encontró un aula libre con dicha capacidad. En caso de encontrarse, se le asignará el aula más chica que cumpla con la capacidad requerida (i.e., la que vaya más justa de acuerdo a lo pedido).
- Para una instancia determinada de nuestro TAD, se desea también poder consultar, para una franja dada, qué aulas están libres. Adicionalmente, se desea poder obtener un ranking de las materias según la cantidad de reservas que hayan efectuado. Por ejemplo, poder obtener el grupo de las 10 materias que más reservas hicieron.

Dada la descripción realizada previamente, se pide definir un TAD AULASINF, considerando las siguientes definiciones:

TAD Franja es String. TAD Aula es String. TAD Materia es String

- (a) Listar qué aspectos son relevantes (y, también, cuáles no) para diferenciar dos instancias del TAD.
- (b) Definir un conjunto de **observadores**. Definir el tipo de cada observador, sus restricciones y dar una mini explicación de cuál es su objetivo. No es necesario axiomatizarlos.
- (c) Definir la **igualdad observacional** para el TAD.
- (d) Definir un conjunto (preferentemente, mínimo) de **generadores**. Definir el tipo de cada generador, sus restricciones y dar una mini explicación de cuál es su objetivo.
- (e) Definir el conjunto de **otras operaciones**, con aquellas que no sean observadores ni generadores y que serán necesarias para responder las preguntas planteadas. Definir el tipo de cada operación y sus restricciones. No es necesario axiomatizarlas.
- (f) Axiomatizar la operación que permita saber qué aulas están libres dada una franja horaria. Para esta operación no está permitido hacer *pattern matching* con los generadores del TAD. Para este punto puede suponer todos los observadores axiomatizados correctamente.

• Ej 1 extender TAD Arbol Binario(α)

α -maximo: $ab(nat) \rightarrow nat$

• maximo(a) \equiv if nil? (a) then 0 } lo correcto sería restringir el dominio con $\{\neg nil?(ab)\}$
else
max3(maximo(izq(a)), raiz(a), maximo(der(a)))
fi

β -Sumar: $ab(nat) \times ab(nat) \rightarrow ab(nat)$ ✓

• Sumar(a, b) $\equiv *$

$* \equiv$ if nil? $(a) \wedge$ nil? (b) then

nil ✓

else if nil? $(a) \wedge \neg$ nil? (b) then

bin(Sumar(nil, izq(b)), raiz(b), Sumar(nil, der(b))) $\equiv b$

else if \neg nil? $(a) \wedge$ nil? (b) then

bin(Sumar(izq(a), nil), raiz(a), Sumar(der(a), nil)) $\equiv a$

else

bin(Sumar(izq(a), izq(b)), raiz(a) + raiz(b), Sumar(der(a), der(b))) ✓

fi

fi

fi

✓

Fin TAD

• Ej 2

2. tieneSustituto(P , abrirTienda) \equiv false $\times \rightarrow p \neq producto(s)$

• tieneSustituto(P_1 , nuevoProducto(S, P_2, n)) \equiv if $P_1 = P_2$ then
false ✓
else
tieneSustituto(P_1, S) ✓
fi

• tieneSustituto(P_1 , compra(S, P_2, n)) \equiv tieneSustituto(P_1, S) ✓

• tieneSustituto(P_1 , venta(S, P_2, n)) \equiv tieneSustituto(P_1, S) ✓

• tieneSustituto(P_1 , defSustituto(S, P_2, sus)) \equiv if $P_1 = P_2$ then
true ✓
else
tieneSustituto(P_1, S) ✓
fi

• Sustituto(P_1 , nuevoProducto(S, P_2, n)) \equiv Sustituto(P_1, S) ✓

• Sustituto(P_1 , compra(S, P_2, n)) \equiv Sustituto(P_1, S) ✓

• Sustituto(P_1 , venta(S, P_2, n)) \equiv Sustituto(P_1, S) ✓

• Sustituto(P_1 , defSustituto(S, P_2, sus)) \equiv if $P_1 = P_2$ then

sus ✓
else
Sustituto(P_1) ✓
fi

• #disponibles(P_1 , nuevoProducto(S, P_2, n)) \equiv if $P_1 = P_2$ then

0 ✓
else
#disponibles(P_1, S) ✓
fi

$\bullet \#disponibles(P_1, compra(S, P_2, n)) \equiv$ if $P_1 = P_2$ then
 $\quad \#disponibles(P_1, S) + n$
 else
 $\quad \#disponibles(P_1, S)$
 fi

$\bullet \#disponibles(P_1, venta(S, P_2, n)) \equiv$ if $P_1 = P_2$ then
 if $\#disponibles(P_1, S) \leq n$ then
 $\quad 0$
 else
 $\quad \#disponibles(P_1, S) - n$
 fi
 else
 $\quad \#disponibles(P_1, S)$
 fi

¿y n'p + el rest. de P2?

$\#disponibles(P_1, fin)) \equiv \#disponibles(P_1, S)$

Ej 2 b

• bajoUmbral(s) \equiv filtrarBajoUmbral(Productos(s), s) ✓

- extendiendo TAD Stock

• filtrarBajoUmbral: $\text{conj}(\text{Productos}) \times \text{Stock } s \rightarrow \text{Conj}(\text{Productos})$

$\{ \{ p: \text{Producto} \mid p \in PS \Rightarrow p \in \text{Productos}(s) \} \}$

• filtrarBajoUmbral(PS, s) \equiv *

o sea, $\{ p \mid p \in \text{Productos}(s) \}$

* \equiv if vacio?(PS) then

! \emptyset ✓

else if tieneSustituto?(dameUno(PS), s) then

if #disponibles(dameUno(PS), s) + #disponibles(sustituto(dameUno(PS), s), s) < \star

$\star < \text{Umbral}(s, \text{dameUno}(PS))$ then

Ag(dameUno(PS), filtrarBajoUmbral(sinUno(PS), s)) ✓

else

filtrarBajoUmbral(sinUno(PS), s) ✓

fi

else if #disponibles(dameUno(PS), s) < $\text{Umbral}(s, \text{dameUno}(PS))$ then

Ag(dameUno(PS), filtrarBajoUmbral(sinUno(PS), s)) ✓

else

filtrarBajoUmbral(sinUno(PS), s) ✓

fi

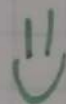
fi

fi

FIN TAD

- #disponibles $S_{\text{ust}}(s, p) \equiv$ if $\text{tiene_sustituto?}(p, s)$ then
 - ! #disponible($\text{sustituto}(p, s), s$) ✓
 - else
 - ! 0 ✓
 - fi

cuando lo más correcto sería que extiendes la restricción con
 $\text{tiene_sustituto?}(p, s)$



Ej 3

a- **Importante** para distinguir dos instancias de AulasInt

Aulas, Franjas Horarias, Materias, Capacidad de las aulas,
Reservas (por materia y franja respecto a un aula)

No importante

Aulas Libres y Ranking de materias por reservas.

b- observadores básicos

• Aulas: AulasInt \rightarrow Conj(Aula)

// Se ven todas las aulas pertenecientes al TAD AulasInt como un conjunto de Aula.

• Franjas: AulasInt \rightarrow Conj(Franja)

// Se ven todas las franjas horarias pertenecientes al TAD AulasInt como un conjunto de Franja.

• Materias: AulasInt \rightarrow Conj(Materia)

// Se ven todas las materias pertenecientes al TAD AulasInt como un conjunto de Materia.

• Capacidad: Aula $a \times$ AulasInt int \rightarrow nat $\{a \in \text{aulas(int)}\}$

// se ve la capacidad de un aula, siempre y cuando el aula pertenezca a las AulasInt

• reservas: Aula $a \times$ AulasInt int \rightarrow dicc(Materia, conj(Franja))
 $\{a \in \text{aulas(int)}\}$

// se ven las reservas realizadas a un aula por cada materia, siempre y cuando el aula pertenezca a las AulasInt

c- $(\text{int}_1, \text{int}_2: \text{AulasInt}) (\text{int}_1 =_{\text{obs}} \text{int}_2 \Rightarrow$ *

* $(\text{franjas}(\text{int}_1) = \text{franjas}(\text{int}_2) \wedge \text{materias}(\text{int}_1) = \text{materias}(\text{int}_2))$

$\wedge \text{aulas}(\text{int}_1) = \text{aulas}(\text{int}_2) \wedge (\forall a: \text{Aula}) (a \in \text{aulas}(\text{int}_1) \Rightarrow$ *

* $(\text{capacidad}(a, \text{int}_1) = \text{capacidad}(a, \text{int}_2)$

$\wedge \text{reservas}(a, \text{int}_1) = \text{reservas}(a, \text{int}_2)) \wedge$

ignoreme XD.

d - Nuevo Int: $\text{dicc}(\text{Aula} \times \text{Capacidad}) \text{as} \times \text{Conj}(\text{Materia}) \text{ms} \times \text{Conj}(\text{Franja}) \text{fs}$

→ AulasInt

$\{ (a_1, a_2 : \text{Aula}) \mid (a, e \text{ claves(as)}) \wedge a_2 \in \text{claves(as)} \}$
 $\Rightarrow \{ \text{obtener}(a_1, \text{as}) = \text{obtener}(a_2, \text{as}) \Leftrightarrow a_1 = a_2 \mid \}$
 $\wedge \neg \exists! (ms) \wedge \neg \exists! (fs)$
 → Estaban bien !!

// Se crea una nueva instancia del TAD definiendo aulas, conjunto a sus capacidades las cuales tienen que ser distintas para todas, materias y franjas. ~~misma~~

• Reservar Aula: $\text{Materia } m \times \text{Franja } f \times \text{nat} \times \text{AulasInt int} \rightarrow \text{AulasInt}$

De última se rechaza automáticamente la reserva.

$\{ m \in \text{materias(int)} \wedge f \in \text{franjas(int)} \wedge$

$\neg \exists (a : \text{Aula}) (a \in \text{aulas(int)} \Rightarrow \exists! t \in \text{obtener}(m, \text{reservas}(a, \text{int})) \}$

// Se pide reservar un aula para una materia m en una franja f para una cantidad de alumnos, solo es posible si esa materia no reservo alguna aula en esa misma franja. Se hace aut por cant alumnos y disponibilidad

e - Aulas Libres: $\text{Franja } f \times \text{AulasInt int} \rightarrow \text{Conj}(\text{Aula}) \setminus \{ f \in \text{franjas(int)} \}$

• Ranking Materias: $\text{AulasInt} \rightarrow \text{Secu}(\text{Material})$

f - Aulas Libres(f, int) $\equiv \text{filtrarAulasLibres}(\text{aulas(int)}, f, \text{int})$

• Filtrar Aulas Libres: $\text{Conj}(\text{Aula}) \times \text{franja} \times \text{AulasInt int} \rightarrow \text{Conj}(\text{Aula})$
 $\{ as \subseteq \text{aulas(int)} \}$

• Filtrar Aulas Libres(as, f, int) \equiv if vacio?(as) then

\emptyset
 else if $f \in \text{Significados}(\text{reservas}(\text{dameUndas}(), f))$
 $\text{filtrarAulasLibres}(\text{sinUndas}, f, \text{int})$

else

$\text{Ag}(\text{dameUndas}, \text{filtrarAulasLibres}(\text{sinUndas}, f, \text{int}))$

fi

fi

Extiende TAD diccionario

- significados: $\text{dicc}(\alpha, \text{conj}(\alpha)) \rightarrow \text{conj}(\alpha)$
- $\text{significados}(d) \equiv \text{significados Aux}(\text{claves}(d), d)$ ✓
- $\text{significados Aux}(cs, d) \equiv$ if $\text{Vacio?}(cs)$ then
 \emptyset
else
 $\text{obtener}(\text{cabeza}(cs), d) \cup$ ✓
 $\text{significados Aux}(\text{sinCabeza}(cs), d)$ ✓
fi
- $\text{significados Aux}: \text{conj}(\alpha)^{cs} \times \text{dicc}(\alpha, \text{conj}(\alpha))^d \rightarrow \text{conj}(\alpha)$ ✓
 $\{cs \in \text{claves}(d)\}$

Fin TAD